(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 September 2003 (25.09.2003)

PCT

(10) International Publication Number
WO 03/079126 A1

(51) International Patent Classification[7]: G05B 19/418, H04L 29/06

(21) International Application Number: PCT/US03/07483

(22) International Filing Date: 12 March 2003 (12.03.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/097,390 14 March 2002 (14.03.2002) US

(71) Applicant: SCHNEIDER AUTOMATION INC. [—/—]; One High Street, North Andover, MA 01845 (US).

(72) Inventor: DUBE, Dennis, J.W.; 8 Birch Lane, Pelham, New Hampshire 03076 (US).

(74) Agent: GOLDEN, Larry, I.; General Patent Counsel, Square D Company, 1415 S. Roselle Road, Palatine, IL 60067 (US).

(84) Designated States *(regional)*: European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEM AND METHOD FOR ACCESSING DEVICES IN A FACTORY AUTOMATION NETWORK

(57) Abstract: A system and method for configuring a resource in a network is disclosed. The system is accessible by a web tool and includes a configuration database being capable of storing a plurality of parameters for configuring the resource. A configuration page also accessible via the web tool, includes a form to facilitate configuration of the resource. The form is capable of receiving a resource parameter entry and a configuration option selection. The configuration option selection and the resource parameter entry compose a URL. A configuration agent is responsive to the URL wherein the agent and the URL cooperate to manage the configuration database.

WO 03/079126 A1

A SYSTEM AND METHOD FOR ACCESSING DEVICES IN A FACTORY AUTOMATION NETWORK

TECHNICAL FIELD

The present invention relates generally to the field of monitoring and controlling input/output modules or devices for a factory automation system. More particularly, the present invention relates to a system and method for accessing objects and components in devices, e.g., programmable logic controllers (PLC), without the use of proprietary protocols.

RELATED APPLICATIONS

This application is a continuation-in-part of U.S. Patent Application Serial No. 08/927,005, filed September 10, 1997, entitled "Web Interface To A Programmable Controller," now U.S. Patent 6,282,454, issued August 28, 2001 **(SAA-1)**. This application is related to the following, commonly assigned applications: "Web Interface To A Programmable Controller," Serial No. 09/738,445, filed December 15, 2000 **(SAA-1-2)**; "Interface To A Programmable Logic Controller," Serial No. 09/223,349, filed December 30, 1998 **(SAA-19)**; "Method For Programming A PLC Using A Web Browser," Serial No. 09/524,171, filed MM/DD/YYYY **(SAA-34)**; "A System For Programming A Factory Automation Device Using A Web Browser," Serial No. 09/635,278, filed MM/DD/YYY **(SAA-34-1)**; "Input Output Module With Embedded Web Server," Serial No. 09/595,159, filed MM/DD/YYY **(SAA-35)**; "A Web Interface To A Device And A Network Control System," Serial No. 09/738,433, filed MM/DD/YYYY **(SAA-35-1)**; "Object Opening Web Enabled Transaction," Serial No. 09/611,996, filed MM/DD/YYYY **(SAA-42)**; U.S. Patent No. 6,061,603, "System For Remotely Accessing An Industrial Control System Over A Commercial Communication Network," issued May, 9, 2000 **(SAA-1-A)**; and "Method and System For Identifying Factory Automation Objects," 09/723,184, filed November 27, 2000 **(SAA-48)**. The contents of these documents are expressly incorporated herein by reference.

BACKGROUND OF THE INVENTION

Methods for remote access of objects and components in a PLC or factory automation device often require unique, proprietary protocols. These methods require custom software to encode, transmit, parse, and interpret the message sent and received within the system. Furthermore, security concerns generally restrict access to the device by requiring transmission through a firewall. In the past, dedicated lines were the common form of communication between such a control system and a remote location. This type of communication had limited application since the control system

was not accessible from multiple locations. While modems have made it possible to access the control system from remote locations, such access is generally restricted to downloading and uploading data files. Often times, a customized interface is required to access the control system by an end user.

With the growth of Internet, and its World Wide Web providing a delivery platform for organizing Internet data through hypertext links, a client-server system can be designed that will give each end user the same type of user friendly interface and universal access to services on the Web. The Web comprises a network of documents, e.g., sites or pages, stored on server computers throughout the world. Each page typically contains text, multimedia offerings, i.e., graphic images, video, or audio; and hypertext links to other web pages or documents. A web tool, e.g., web browser, allows a user to read and interact with the web page. The browser is a graphical software program that sends commands to the Internet Web site and displays available page information. Several browsers are commercially available from different manufacturers.

Most personal computers or work stations can be used by an end user to connect to the Web through the commercially available browsers. Communication over the Internet and other networks requires one of several available protocols. Protocols such as Internet Protocol (IP) provide for file transfers, electronic mail, and other services. Commercially available programming languages such as Java, along with Hypertext Markup Language (HTML), are used in designing layouts and graphics for a web site or page and have extended Internet technology such that a web site can be used for dynamic applications, e.g. applets, that can be downloaded and run by the end user.

Programmable logic controllers (PLCs) are widely used in industry and process control. Many manufacturers provide factory automation information using Microsoft Windows and other types of communication networking environments. These networks are usually slow, not universally accessible, and are limited to monitoring and data exchange. Specialized industrial networks using proprietary fieldbus alternatives can be very expensive. Conversion products are required to allow information carried over those networks to be visible on a general purpose network. There are significant installation and other deployment costs associated with the existence of such intermediate devices. Firewalls between the Web server and the application are designed to solve problems of security and are not designed for high performance.

It is desirable to develop a system whereby a user can utilize general purpose networks, such as the Internet and specialized industrial networks, directly connected to input/output devices for remote monitoring and control of input/output modules or devices.

This invention is designed to solve these and other problems.

SUMMARY OF THE INVENTION

An embodiment of the present invention is directed to a system for configuring a resource in a network. The resource being operably connected to a web tool providing access to the network. The system comprises a configuration database having a plurality of parameters for configuring the resource. A configuration page includes a form to facilitate configuration of the resource. The configuration page is accessible by the web tool and capable of receiving a resource parameter entry and a configuration option selection. The configuration option selection and the resource parameter entry compose a magic-URL. The configuration agent includes an agent API responsive to the magic-URL wherein the configuration agent and the magic-URL cooperate to manage the configuration database.

A further aspect of the present invention includes a configuration component having a component API. The configuration component API is operably connected to the configuration agent wherein the component API, configuration agent, and configuration database cooperate to configure the resource.

Another aspect of the present invention includes a remote procedural call mechanism to facilitate communication between the web tool and the configuration agent.

Yet another aspect of the present invention includes a configuration component web page. The configuration component web page is created in response to the magic-URL and the configuration database.

Another embodiment of the present invention is directed to a method for configuring a network resource. The network includes a configuration database and a configuration agent wherein a web tool provides access to the network. The method comprises providing a resource configuration page communicable with the web tool. The resource configuration page has a form for facilitating configuration of the resource. The form is capable of receiving a resource parameter entry and a configuration option selection. The configuration option selection and the resource parameter entry compose a magic-URL. The magic-URL is sent via the web to and received at the configuration agent. In response to the magic-URL, a configuration API is selected and executed. The configuration API cooperates with the magic-URL to manage the configuration database.

A further aspect of the present invention provides a network configuration page comprising a link to the resource configuration page. The network configuration page includes information about the network.

Yet another aspect of the present invention is directed to maintaining a

configuration database page in response to the transmitted magic-URL. The configuration database page contains information about the configuration database and is transmitted to the web tool. The network configuration page is maintained to reflect the configuration of the database.

An object of the present invention is to utilize industry standard protocols for providing an interface to a resource, e.g., device, in a network.

Another object of the present invention is to reduce the need for proprietary client-server protocols for accessing a network resource.

A further object of the present invention is to utilize industry standard protocols for configuring a network resource.

Yet another object of the present invention is to provide a web interface for on-line configuration of a controller.

Other features and advantages of the invention, which are believed to be novel and nonobvious, will be apparent from the following specification taken in conjunction with the accompanying drawings in which there is shown a preferred embodiment of the invention. Reference is made to the claims for interpreting the full scope of the invention that is not necessarily represented by such embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 depicts a block diagram of one embodiment of the present invention;

FIG. 2 depicts a block diagram of one embodiment of the present invention utilizing an RPC mechanism;

FIG. 3 is a block diagram of FIG.2 wherein the RPC mechanism is SOAP/XML;

FIG. 4 is a block diagram of one embodiment of the present invention depicting the utilization of the magic-URL;

FIG. 5 is a block diagram depicting one embodiment of the interface module operably connected to the CPU and web tool;

FIG. 6a is a flow chart representing transitions of one embodiment of the present invention;

FIG. 6b is a table representing the behavior of one embodiment of the configuration agent in accordance with the transitions shown in FIG. 6a;

FIG. 7 is a block diagram depicting an alternative embodiment of the present invention;

FIG. 8 is one embodiment of the form utilized to configure the configurable resource;

FIG. 9 is one embodiment of the magic-URL;

FIG. 10 is a table comprising a sample of configuration agent APIs;

FIG. 11 is one embodiment of the code architecture utilized in the preferred

embodiment of the present invention;

FIG. 12 depicts one embodiment of a configuration resource page for an address server; and,

FIG. 13 depicts one embodiment of an address server configuration database file.

FIG. 14 depicts one embodiment of a web page for facilitating changes to the address configuration file; and,

FIG. 15 depicts one embodiment of an address server node configuration web page linked to the address server configuration page of FIG. 14.

DETAILED DESCRIPTION

Although this invention is susceptible to embodiments of many different forms, a preferred embodiment will be described and illustrated in detail herein. The present disclosure exemplifies the principles of the invention and is not to be considered a limit to the broader aspects of the invention to the particular embodiment as described.

The present invention discloses a method and a system for configuring a network resource 10. The network and many of its components are accessible via a web tool 12, e.g., browser. A configuration page 16 includes a form 18 to facilitate configuration of the resource 10. The form 18 is accessible via the web tool 12 and capable of receiving a resource parameter entry 20 and a configuration option selection 22. A configuration agent 24 is responsive to the form 18 and is capable of interfacing with the resource 10. The configuration agent 24 coordinates the configuration of the resource 10. An agent API 26 is associated with the selected configuration option 22. Dependent upon the entered resource parameter 20 and the selected configuration option 22, a configuration database 28 is accessed and utilized for configuring the resource 10.

An alternative embodiment of the present invention comprises an interface module 14 operably connected between the resource 10 and the browser 12. The configuration agent 24 is operably connected to the interface module 14 and coordinates the configuration of the resource 10. The configuration database 28 is accessible via the interface module 14 and maintains a plurality of parameters for configuring the resource 10. An industry standard remote procedural call mechanism 30 facilitates communication between the web tool 12 and the interface module 14. The configuration database 28 may reside as a binary structure or ASCII text file.

The configurable resource 10 is defined by the configuration database 28. Some databases are simple, e.g., an operating mode indicator, and others are more complex, e.g., SNMP MIB. The configuration database 28 can be created by an ASCII editor and downloaded to the system via FTP. Some types of configuration databases 28 are files

or components such as: global data variables, controllers, DHCP server databases, and bandwidth setting managers. The scope of a configuration database 28 indicates the parts of the system architecture affected by the database. Affected parts can include system components such as: communication ports, functional components or APIs, and abstractions such as: object modules, address spaces, and name spaces. These components provide details about global data variables such as DataID, symbol name, StateRAM address, and its length in response to a device manager's 30 configuration request.

The configuration agent 24 is interfaced between the browser 12 and the configurable resource 10. FIG. 1. The configuration agent 24 enables the web-based interface 14 to configure the network component 10. The configuration agent 24 is utilized to communicate with the browser 12 for providing an application program interface (API) 26 for managing the configurable resource 10. The configuration agent 24 also retrieves selected parameters stored in the configuration database 28.

To execute the configuration agent API 26, the API is operably selected via the browser 12. The browser can utilize Java, GUI, or any other user communication interface mechanism. For example, FIG. 2 depicts a remote procedure call mechanism (RPC) 32 including a client module 34 and a server module 36. The user defines the parameters of the API 26 and initializes its execution. The client module 34 of the RPC mechanism 32 encodes and transmits the API request to the server module 36 of the RPC mechanism 32. The server module 36 decodes the API request and makes an API method call. The configuration agent 24 executes the API 26 and returns resulting data to the server module 36 of the RPC mechanism 32. The server module 36 of the RPC mechanism 32 encodes the API response and sends it to the client module 34 of the RPC mechanism, thus completing the execution of the API 26. The client module 34 of the RPC mechanism 32 presents the selected return data to the user through the browser 12. Some alternative embodiments of the RPC mechanism 32 include: Java/JMI, MS DCOM, CORBA/ORBs, and SOAP/XML. For example, an embodiment utilizing SOAP/XML is shown in FIG. 3. SOAP defines method requests and responses, parameter and return value data types, and method invocation semantics. XML encodes the SOAP requests and responses. HTTP transmits the XML-encoded SOAP request and responses between the client 34 and the server 36. Both the client and the server must have XML parsers and generators.

The magic-URL 38 is a CGI-like mechanism utilized to access the configuration agent API 26. The magic-URL 38 is a link to a specialized function, i.e., API 26, rather than to a web page. FIG. 4. To accomplish a configuration agent API call, the configuration data page 16 is accessed via the browser 12. The form 18 containing

various fields for entering parameter variables 40 and selecting configuration options 42 resides at the configuration web page 16. Desired variables 40 and configuration options 42 are selected on the form 18 and submitted via the browser 12. The contents 40, 42 of the form 18 are encoded into a command composing the magic-URL 38. The selected API configuration option 42 is identified by a tag/value pair, e.g., API=Gda_update — this particular API example updates an entire global database using input selected from the configuration page and encoded in the magic-URL 38. The interface module 14 utilizes the incoming magic-URL 38 to call the API 26. The API 26 uses the URL-encoded data as input and executes the associated function. Preferably, a database web page is either created or updated to reflect the current contents of the database 28.

In another embodiment of the present invention, a configuration component 11 provides data exchanges for configuration of a resource, e.g., controller. The configuration component 11 must be configured to determine the location and the quantity of application variables to be exchanged in each station. The interface module 14 can be configured via the browser 12 or FTP. To achieve proper configuration of the configuration component 11, a configuration file containing configuration parameter agents 24, e.g., glbdat.ini, cooperates with the database 28. FIG. 5.

The configuration component 11 includes an agent that implements a resource protocol and offers an API for the configuration sequence used by the GLBD_Configure function of the configuration component 11. The configuration component agent 11 is utilized twice; once when the configuration of the service is set, and again during the configuration of the resource 10. i.e., the configuration database 28 provides the appropriate information to the configuration component 11 using a selected configuration agent API.

FIGS. 6a and 6b depict the behavior of the configuration agent 24 according to a device manager 30 request and the associated states of the component state machine. The configuration component 11 provides the GLBD_Configure API. Once called by the device manager 30, this function will configure the resource 10 with the information contained in the database 28. To access this information, the configuration component 11 will use the API 26 provided by the configuration agent 24. FIG. 7.

Another simple example of the present invention includes the user connecting to the web interface module 14 via the browser 12 to configure the resource component 10. The configuration agent 24 executes the selections on the screen and stores the configuration chosen by the user in a file, i.e., glddata.ini. At this time, the selected entries of the configuration database 28 are not set in the resource component 10 — only the parameters are stored in the database 28. During configuration of the resource

component 10, the configuration agent 24 provides the component 10 with the information stored in the glbdata.ini file of the database 28 to properly achieve its configuration. The glbdata.ini file is managed by the configuration agent 24 and can be edited manually by the user and then downloaded via FTP to the web interface module 14. The configuration agent 24 creates and manages the database 28. However, it is possible for the user to create the database from the ASCII editor and to download it to the web interface module 14. The browser 12 communicates with the configuration agent 24 to manage the database file 28. FIG. 8. The functions provided by the configuration agent 24 are accessible through the browser 12 and the component 10. The configuration agent 24 reads the database 28 then forwards the data to the component 10 in order to complete its GLB_Configured function. The format of the entry is defined in order to store each parameter accessible by the user from the web configuration page 16 — this information is consistent with the needs of the GLB_Configure function.

The configuration page 16 is preferably written in HTML and may contain JavaScript functions. The web page 16 allows the user to define variables and global data configuration parameters, i.e., group address, multicast filtering, distribution period, health time out, health bit address, start address, etc. For each variable entry, the user may specify parameters, e.g., type (publish/subscribe), symbol, address (the location of the variable in the state RAM or the CPU), and length. The user edits a form 44 and clicks the "Update Symbols" button. This causes the contents of the form 44 to be sent to the agent 24 using the HTTP Get method. The contents of the form 44 are encoded into the magic-URL 38.

Initially, the database 28 is blank. To enable a correct initialization sequence of the component 10, the user must populate or create the database 28. The database 28 can be created with the browser 12 or FTP. The user accesses the configuration page 16, edits the form 44, and submits the contents of the form to create the first version of the database 28. Alternatively, the user can download an existing database into the web interface module 14 via FTP. The syntax must be checked prior to ensure that failures do not occur during initialization.

The exchange of data between the configuration agent 24 and the browser 12 utilizes the magic-URL 38. Essentially, the database 28 is modeled as an object and the configuration agent 24 provides a set of methods to manage the object. The content of the current database 28 can be accessed by the user via an HTML file created in response to a request. To send the new values chosen by the user to the agent 24, the content of the form 44 is sent using the HTTP GET method, i.e., data is encoded at the end of the magic-URL 38. The magic-URL 38 enables the call of a specific function 26,

API . Unlike a typical URL, an HTML page is not located at the address referenced within the magic-URL 38, but rather, the magic-URL points, or triggers a call to a function. A preferred embodiment of the data exchange between the user and the configuration agent incorporates HTTP wherein the user's choices and configuration parameters are contained in an encoded string located at the end of the magic URL. An example of a magic-URL is shown in FIG. 9. The magic-URL comprises one continuous character string; however, for descriptive purposes, carriage-returns have been inserted within the string for easier readability.

The information encoded within the magic-URL 38 is parsed by the function call 26 identified within the magic-URL. The contents of the magic-URL reflect the parameters and variables chosen by the user from the form web page 18. The magic-URL 38 appears to function as a typical CGI mechanism because an HTML file can be created in response to the activity of the API 26, e.g., updating a database, wherein the contents of the updated database are formatted and sent to the browser 12 for display.

Configuration agent APIs 26 provide mechanisms for creating programs for managing the database 28 and to retrieve data previously stored. Sample configuration agent APIs 26 are shown in FIG. 10. Referring to FIG. 11, the preferred code architecture utilizes three files to access the global data agent. These files are: cfgagent.cpp, vxwmisc.c, and vxwhttp.c. File cfgagent.cpp is the core of the agent API code and implements all of the API of the configuration agent, i.e., open, read, save, etc. File vxwmisc.c interfaces with the web tool. The function, Gda_IF is called when the magic-URL 38 is accessed. Then, according to the query string encoded within the magic-URL 38, the appropriate function is called in the cfagent.cpp. The creation of the magic-URL 38 — the link to Gda_IF function within vxwmisc.c — is established within vxwhttp.c.

Another embodiment of the present invention is shown in FIG. 12. The configurable resource 10 is an address server, preferably DHCP. The DHCP server database is capable of storing and retrieving entries dynamically added to the database 28. This functionality is particularly effective in coordinating the replacement of a faulty resource 10 on the network. The configuration parameters of the address server 10 are located in a file, addserv.ini. This file can be created or modified using the web configuration page 16 or an FTP client to upload/download the addreserv.ini file. FIG. 13.

At system start-up, the existence of a configuration file 28 is verified. If the address configuration file 28 is not present, one is created. The address configuration file 28 contains the configuration parameters 40 of the DHCP server 10. These

parameters include: a role name or MAC address; an IP address; a gateway; and a subnet mask. Preferably, two web pages are utilized to configure the address server 10. A first page 46 depicts the device or devices being utilized and their respective configurations. FIG. 14. A second page 48 facilitates modifying, or editing, the device's configuration and is accessible via a link from the first page. FIG. 15.

From the address configuration page 46, a new device 10 can be accessed. An entry can be selected by clicking on the associated radio button to the left of the entry. When the entry is selected, all the device information, i.e., role name, IP address, etc. is placed in a hidden field. This information is stored in an array during the creation of the address configuration web page for use by JavaScript.

When adding an entry, the entry is selected by clicking the radio button and the address server node configuration page 48 will appear. If a device was selected, the configuration for that device will appear on the page. Otherwise, default values are presented on the page. To change an entry, the user is allowed to modify the configuration of the device via the address server node configuration page reflecting the current configuration that appears in response to the selection of the entry.

To remove an entry, the user selects the entry and then selects the "Delete an Entry" icon. The selected entry will be removed from the database and the page redisplayed to reflect this change. When the user has successfully added, modified, or removed an entry, confirmation of this action will be provided on the screen; via a generated web page.

Generally, the address server node configuration web page 48 is utilized to add or modify an entry in the database. In either case, all fields except for Role Name and MAC Address must be filled. If the MAC address field is defined, the Role Name field is empty; and if the Role Name field is defined, the MAC address field will be empty. The fields include: Entry Number (id); Role Name(rn) — each role name must be unique; Device IP Address (ip); Device MAC Address (ha) in hexadecimal; Submet Mask (sm) in IP format; and Gateway (gw) — preferably on the same subnet as the device and in IP format.

To effect the change, the address server configuration file 46 must be updated on the server side. To do this, the address server configuration agent 24 is utilized to update the address server configuration file 28 with the parameters provided by the user through the address server configuration web pages 46, 48. The configuration agent 28 utilizes the magic-URL 38 encoded with the information from the form 18 populated by the user. The type of API tasks performed by the agent 28 include:

ADD: add a device in the database and the system;

CHANGE: change the configuration of a device in the database and the

system;

DELETE: remove a device from the database and the system; and,

REFRESH: display the configuration page with the current configuration.

In response to the type of task requested to be performed, an appropriate API is selected and accompanied with the appropriate data required for executing the task. Not all tasks require the same data. For instance, ADD requires Role Name, IP Address, MAC Address, Subnet Mask, and Gateway, while DELETE only requires Entry Number; and REFRESH does not require any data. CHANGE requires everything that ADD requires and an Entry Number.

A query string comprises the magic-URL 38 and information required for executing the function 26 called by the magic-URL . An example of the magic-URL is: xxx/secure/embedded/DHCPa_IF?API=CHANGE&id=0&rn=myEIO&ip=139.158.13.222 &ha=FF+FF+FF+FF+FF+FF&sm=255.255.255.0gw=139.158.8.1&submit=Submit+the+ Form, wherein xxx/secure/embedded/DHCPa_IF? is the function call with the data following thereafter.

Configuration information entered at the configuration web page can include any combination of the following parameters, including others: id (entry number); rn (role name); ip (device IP address); ha (device MAC address); sm (subnet mask); gw (gateway); and submit (caption of the form submit button).

The relationship of the configuration database 28 to existing control applications is of particular importance. Preferably, the definition of the database 28 resides in a single location. Due to design architecture, some devices/systems utilize proxies or replications as an intermediate interface to the configuration database 28. If such duplication is implemented, it is preferable that a mechanism exist to ensure automatic and timely synchronization of the configuration database 28 definition with any cached or replicated definition. Often, this synchronization can occur in real-time. As noted earlier, when a configuration is written to the interface module 14, the content of the file is not automatically synchronized with the running component. The resynchronization must be accomplished by some other mechanism, such as resetting the resource. If dynamic changes to the runtime version of the database are allowed — which is really a replication of the original database — it is imperative to ensure that a mechanism exist to make the persistent version consistent with the runtime version. Conversely, if dynamic changes to the persistent version of the database 28 are allowed, a mechanism should be provided to resynchronize the resource 10 with the persistent version of the database.

From the foregoing, it will be observed that numerous variations and modifications may be effected without departing from the spirit and scope of the

invention. It is to be understood that no limitation with respect to the specific apparatus illustrated herein is intended or should be inferred. It is, of course, intended to cover by the appended claims all such modifications as fall within the scope of the claims.

CLAIMS

We claim:

1. A system for configuring a resource in a network, the network being accessible via a web tool, the system comprising:

a configuration database being capable of storing a plurality of parameters for configuring the resource;

a configuration page being accessible via the web tool and having a form to facilitate configuration of the resource, the form being capable of receiving a resource parameter entry and a configuration option selection;

a magic-URL comprising the configuration option selection and the resource parameter entry; and,

a configuration agent being operably responsive to the magic-URL wherein the agent and the magic-URL cooperate to manage the configuration database.

2. The system of Claim 1 further comprising:

a remote procedural call mechanism for facilitating communication between the web tool and the interface module.

3. The system of Claim 1 further comprising:

an interface module being operably connected between the resource and the web tool, the interface module being accessible via the magic-URL.

4. The system of Claim 1 wherein the configuration agent includes an agent API.

5. The system of Claim 1 further comprising:

a configuration component having a component API and being operably connected to the configuration agent wherein the component API, configuration agent, and configuration database cooperate to configure the resource.

6. The system of Claim 1 further comprising a configuration component web page, the configuration component web page being created in response to the magic-URL and the configuration database.

7. The system of Claim 1 wherein the resource is a global configuration component.

8. The system of Claim 7 wherein the configuration database includes a resource parameter selected from the group consisting of a type field, a symbol field, a network address field, and a length field.

9. The system of Claim 1 wherein the resource is an address server database.

10. The system of Claim 9 wherein the configuration database includes a resource parameter selected from the group consisting of a role name field, a MAC address field, an IP address field, a subnet mask field, and a gateway field.

11. The system of Claim 1 wherein the resource is a DHCP server database.

12. The system of Claim 1 wherein the resource is a bandwidth management component.

13. The system of Claim 2 wherein the remote procedure call mechanism comprises Java/JMI.

14. The system of Claim 2 wherein the remote procedure call mechanism comprises MS DCOM.

15. The system of Claim 2 wherein the remote procedure call mechanism comprises CORBA/ORBs.

16. The system of Claim 2 wherein the remote procedure call mechanism comprises SOAP/XML.

17. The system of Claim 1 wherein the configuration database is embedded within the interface module.

18. A method for configuring a network resource comprising the steps of:

providing a resource configuration page communicable with a web tool, the configuration page including a form for facilitating configuration of the resource, the form being capable of receiving a resource parameter entry and a configuration option selection;

receiving a magic-URL, the magic-URL being responsive to the resource

configuration page and comprising the configuration option selection and the resource parameter entry;

selecting a configuration API in response to the magic-URL; and,

configuring the resource in response to the configuration API.

19. The method of Claim 18 further comprising the step of:

providing a network configuration page comprising a link to the resource configuration page, the network configuration page having information about the network.

20. The method of Claim 18 further comprising the steps of:

maintaining the configuration database to reflect the configuration of the resource; and,

creating a configuration database page for providing information about the configuration database, the configuration database page being accessible via the web tool.

21. The method of Claim 20 further comprising the step of:

maintaining the network configuration page to be representative of the configuration database.

22. A method for configuring a network resource comprising the steps of:

providing a resource configuration page being accessible by a web tool and having a form for facilitating configuration of the resource, the form being capable of receiving a resource parameter entry and a configuration option selection;

constructing a magic-URL in response to the resource configuration page, the magic-URL comprising the configuration option selection and the resource parameter entry;

selecting a configuration API in response to the magic-URL; and,

configuring the resource in response to the configuration API.

23. The method of Claim 22 further comprising the step of:

providing a network configuration page comprising a link to the resource configuration page, the network configuration page providing information about the network.

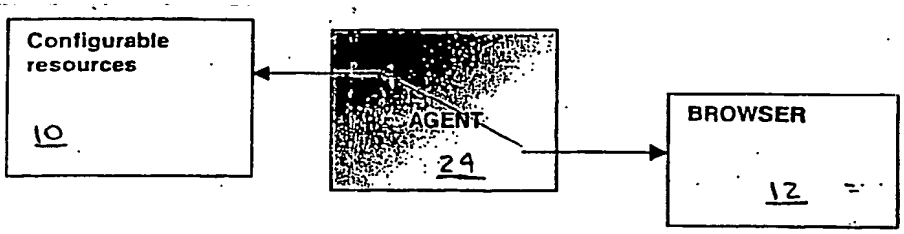
Configurable resources
10
AGENT
24
BROWSER
12

FIG. 1

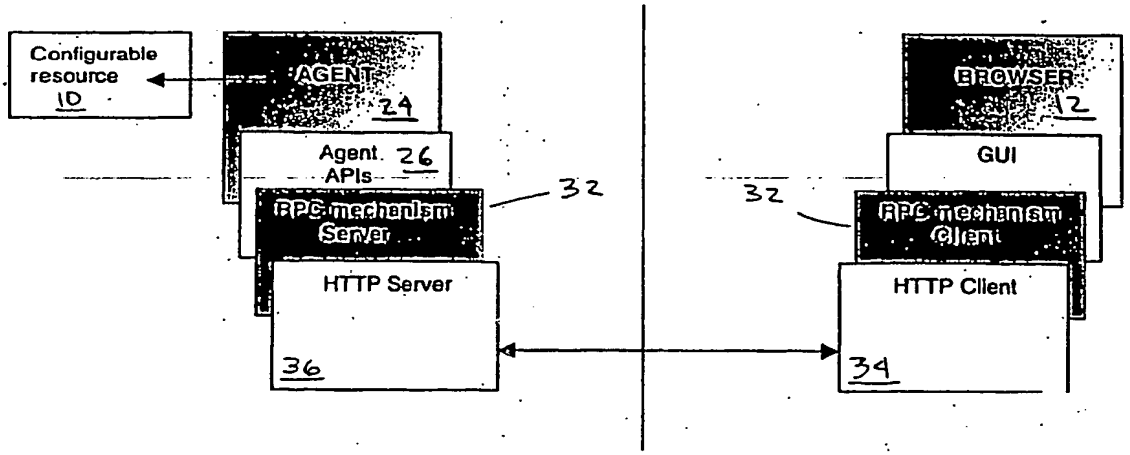
Configurable resource
10
AGENT
24
Agent APIs
26
RPC mechanism Server
32
HTTP Server
36
BROWSER
12
GUI
32
RPC mechanism Client
HTTP Client
34

FIG. 2

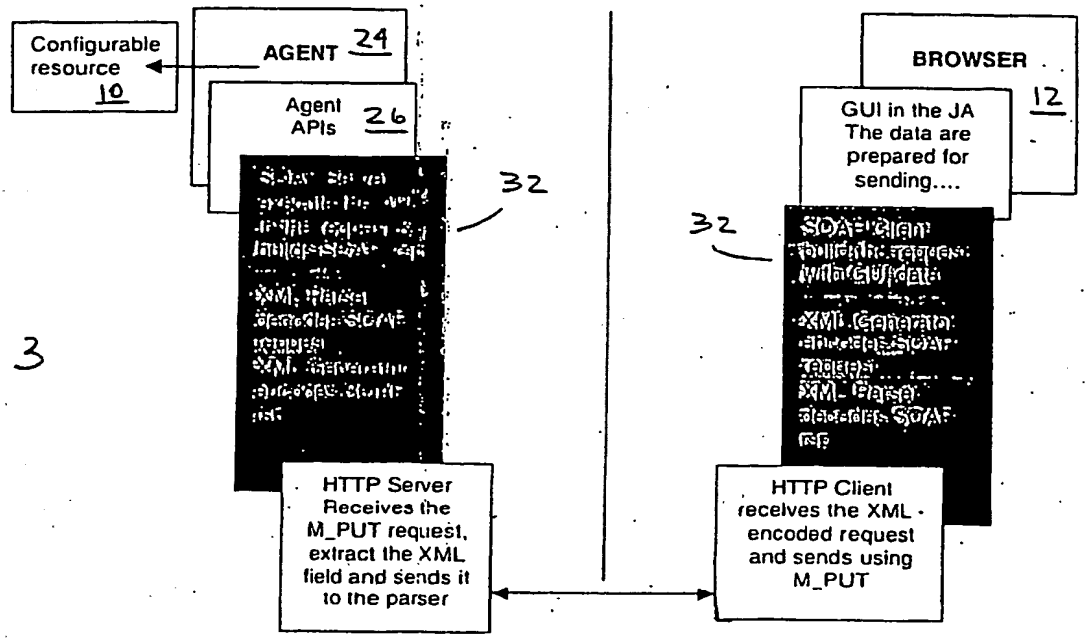
Configurable resource 10
AGENT 24
Agent APIs 26
32
BROWSER
12
GUI in the JA The data are prepared for sending....
32
HTTP Server Receives the M_PUT request, extract the XML field and sends it to the parser
HTTP Client receives the XML - encoded request and sends using M_PUT

FIG. 3

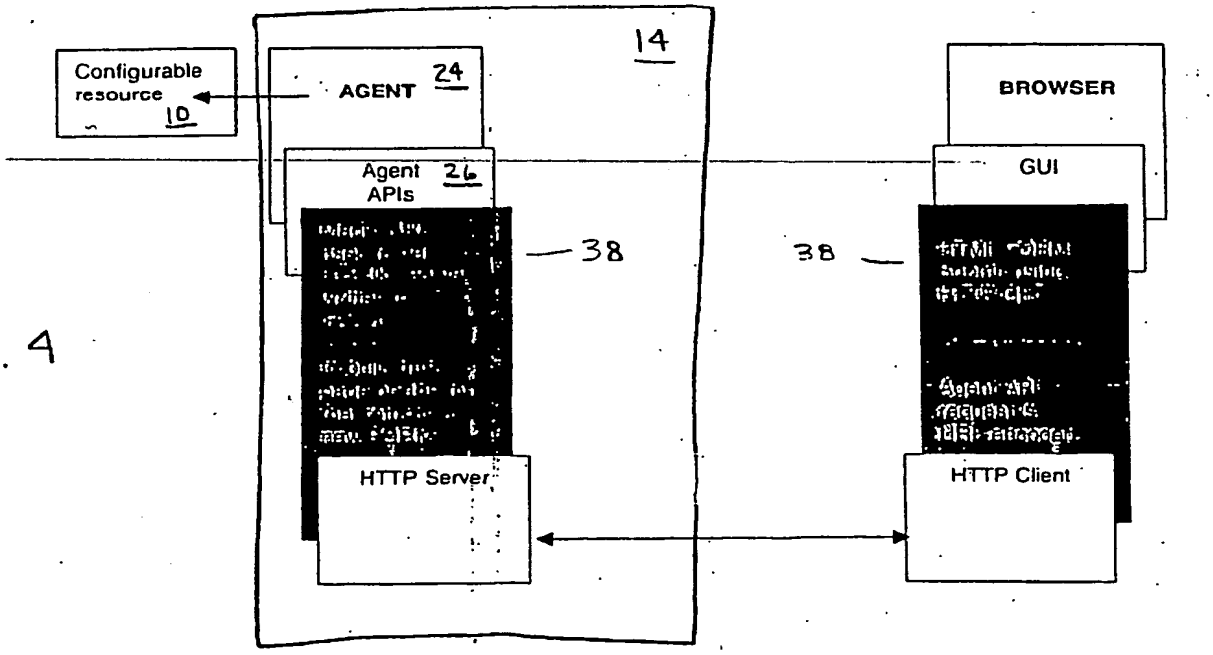
14
Configurable resource 10
AGENT 24
Agent APIs 26
38
BROWSER
GUI
38
HTTP Server
HTTP Client

FIG. 4

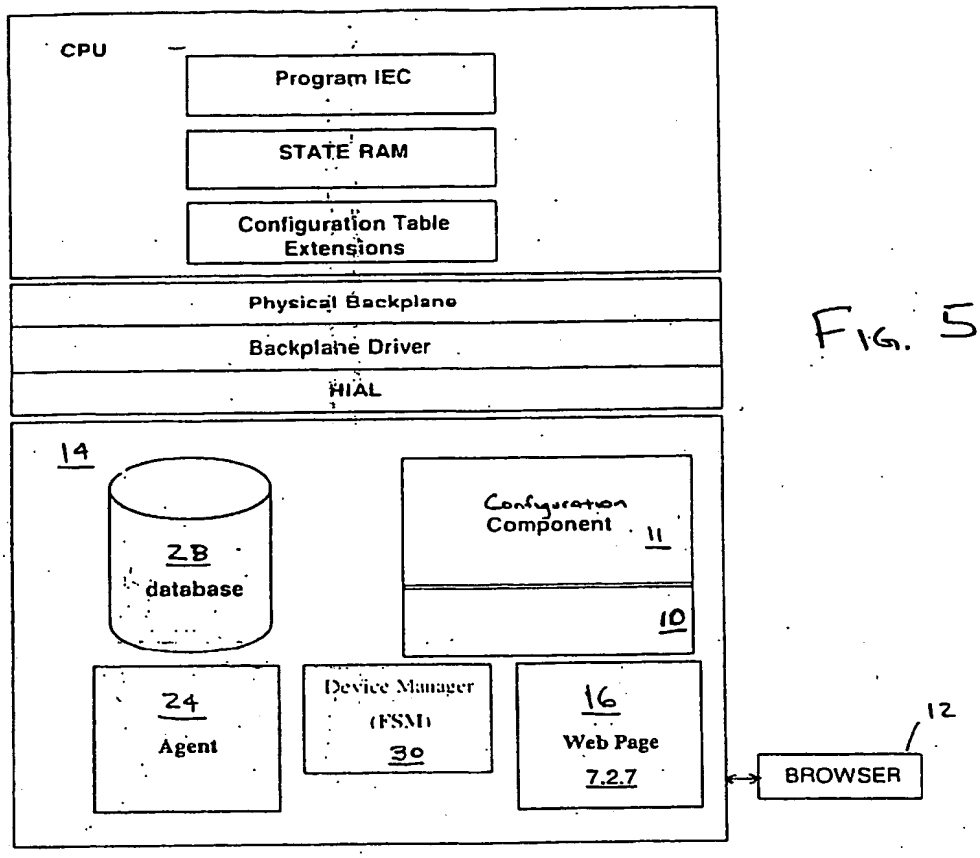
CPU
Program IEC
STATE RAM
Configuration Table Extensions
Physical Backplane
Backplane Driver
HIAL
14
28
database
Configuration Component
11
10
24
Agent
Device Manager (FSM)
30
16
Web Page
7.2.7
12
BROWSER

FIG. 5

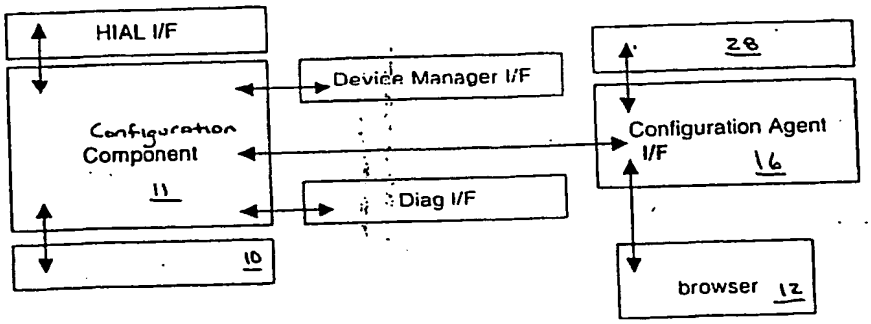
HIAL I/F
Device Manager I/F
Configuration Component
11
Diag I/F
10
28
Configuration Agent I/F
16
browser
12

FIG. 7

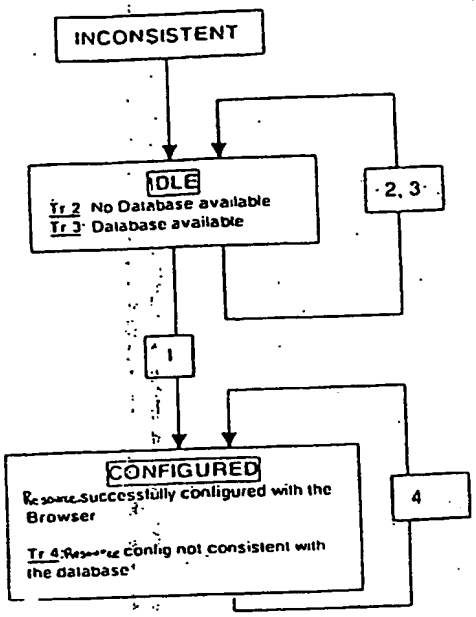

Fig 6a

Transition number	Current State of the Device Manager	Event	Database Status	Action related to the configuration agent	Next State of the device manager
1	IDLE	GLBD_Configure	- Available - Consistent with the current component configuration	Get parameters from the database	CONFIGURED
2	IDLE	GLBD_Configure	Not available or not correctly filled out (i.e.: if the user uses an ASCII editor)	The Heading of the database failed	IDLE
3	IDLE	Web_Event Update Symbols request	Overwritten	Overwrite the Database	IDLE
4	CONFIGURED	Web_Event Update Symbols request	Overwritten (no more consistent with the component)	Overwrite the Database	CONFIGURED

Fig. 6b

FIG. 8

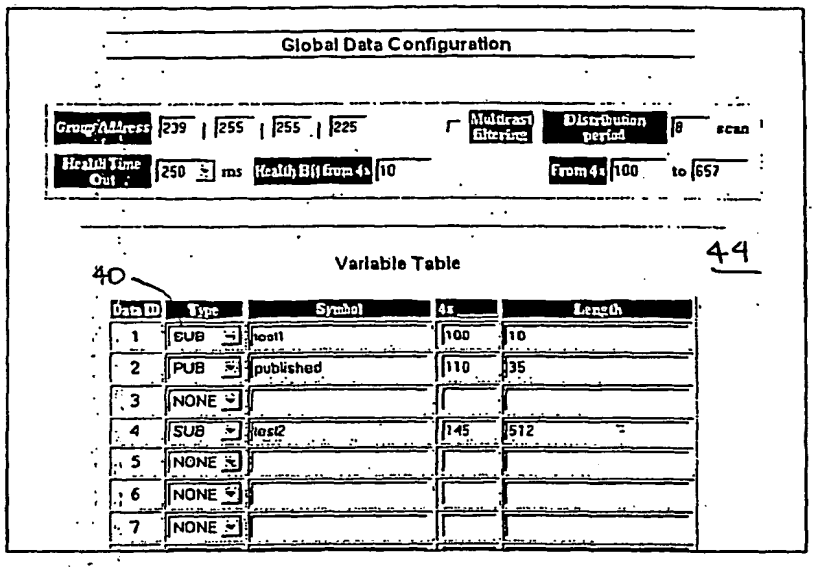

FIG. 9

38

```
http://elo19/secure/embedded/GDa_IF?
API=GDa_update&
reqgroupAddressA=238&reqgroupAddressB=255&reqgroupAddressC=255&reqgroupAddressD=255&
reqdistributionPeriod=2&
reqhealthTimeOut=300&
reqhealthBit=10&
reqzoneFrom=500&a0=725S
t1=SUB&s1=BlueSky&a1=500&l1=328
t2=SUB&s2=Second area&a2=532&l2=64&
t3=NONE&s3=&a3=&l3=&
```

FIG. 10

26

Function	Description	Comments
GDa_emptyDb	Empty the current database (create a new one if none exists)	This function is called by the GDa_readGlbdFile() function when it is not enable to open a previous file
GDa_update	Store the new parameters set by the user from the Web page in the file and / or send the current configuration to the browser	Sending the current parameters is mandatory once the update function is called
GDa_buildHTMLresponse	Build a HTML page to display to the user according the database	Called by the GDa_update function
GDa_readGlbdFile()	Read the parameters in the configuration file	Performs some test to be sure that the parameters have the good format
GDa_parseURL()	Read the parameters from the URL	
GDa_writeGlbdFile()	Write the parameters in the file	
GDa_getHealthBitAddress	Return the Health bit address	
GDa_getGlbDataConf	Retrieve the configuration stored in the file, without the Health bit address	Store the configuration in the GDQuantum structure

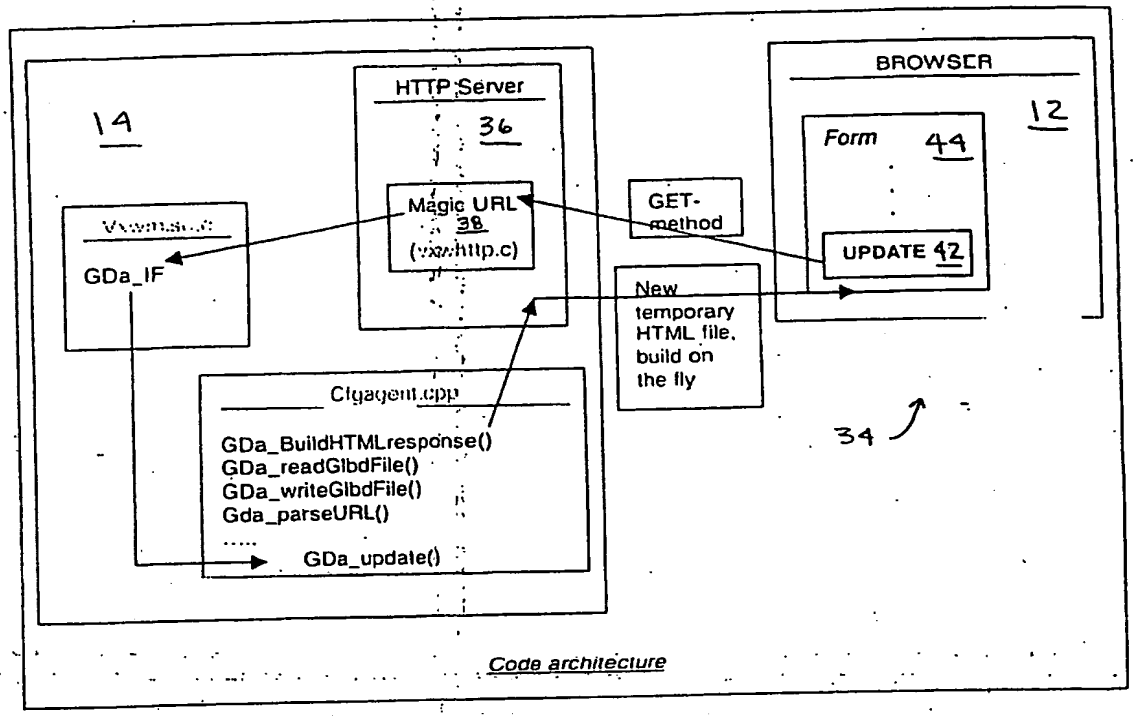
14
HTTP Server
36
Magic URL
38
(vw-http.c)
GDa_IF
BROWSER
12
Form
44
UPDATE 42
GET-method
New temporary HTML file, build on the fly
Cfgagent.cpp
GDa_BuildHTMLresponse()
GDa_readGlbdFile()
GDa_writeGlbdFile()
Gda_parseURL()
.....
GDa_update()
34
Code architecture

Fig. 11

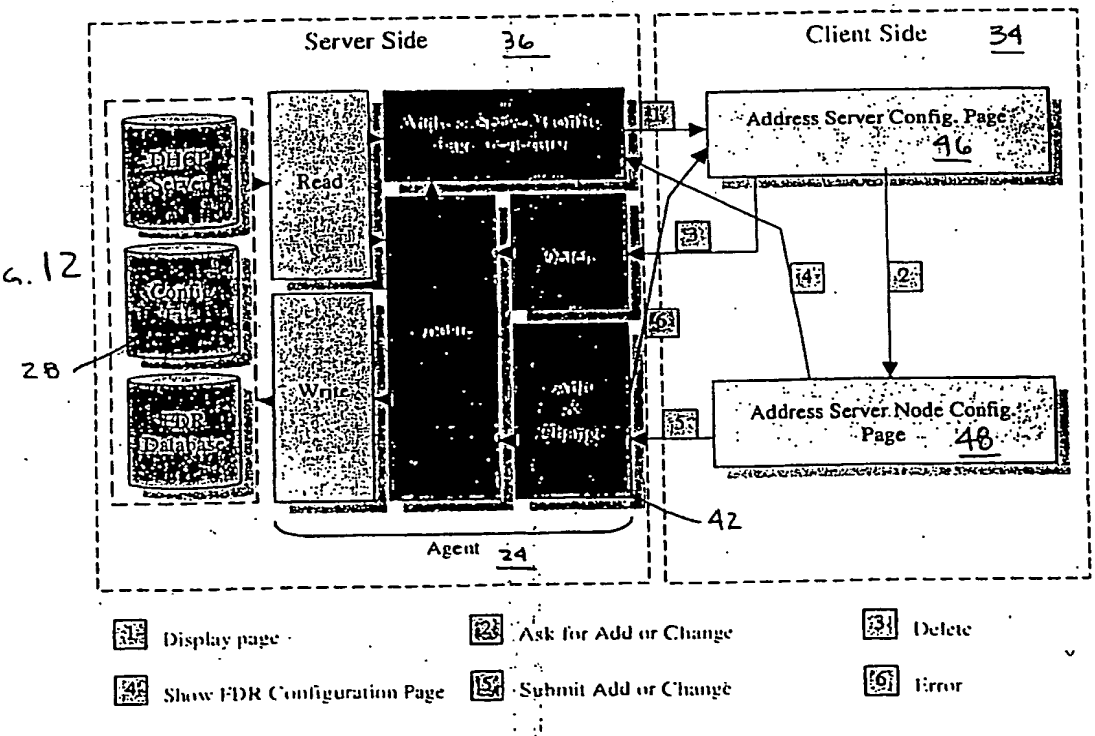
Server Side 36
Client Side 34
Read
Write
Address Server Config. Page 46
Address Server Node Config. Page 48
28
42
Agent 24
Display page
Ask for Add or Change
Delete
Show FDR Configuration Page
Submit Add or Change
Error

Fig. 12

28

```
# Database for FDR server
#
# ========================
# path: /wwwroot/conf/dhcp/
# file: addrserv.ini
# ========================
#
# Blank lines and lines beginning with '#' are ignored.
#
# ========================
#
#       One entry by line
#
#       rn -- role name (must have 16 or less characters)
#       gw -- gateways
#       ha -- hardware address
#       ip -- device IP address
#       sm -- subnet mask
#
# Example:
# rn=myName:ip=192.168.5.125:sm=255.255.255.0:gw=192.168.1.1:ha=FF00AB15B251
#
# ========================
#

rn=myEIO:ip=139.158.13.222:sm=255.255.255.0:gw=139.158.8.1:ha=
rn=:ip=139.158.13.225:sm=255.255.255.0:gw=139.158.8.1:ha= FF00FF00FF00
rn=EIOTest:ip=139.158.13.125:sm=255.255.255.0:gw=139.158.8.1:ha=
```

FIG. 13

46

Address Server Configuration

	Role Name	Mac Address	IP Address	Subnet Mask	Gateway	
○	myEIO		139.158.13.222	255.255.255.0	139.158.8.1	
○		FF 00 FF 00 FF 00	139.158.13.21	255.255.255.0	139.158.8.1	
○	EIO		139.158.13.222	255.255.255.0	139.158.8.1	
○	*too_Long_Role_n#*		*139.158.13.125*	*255.255.255.0*	*139.158.8.1*	!

Refresh Address Server Database Table

Add a New Entry | Change an Entry | Delete an Entry

42

Home | Configure NOE | NOE Properties | NOE Diagnostics | Support

FIG. 14

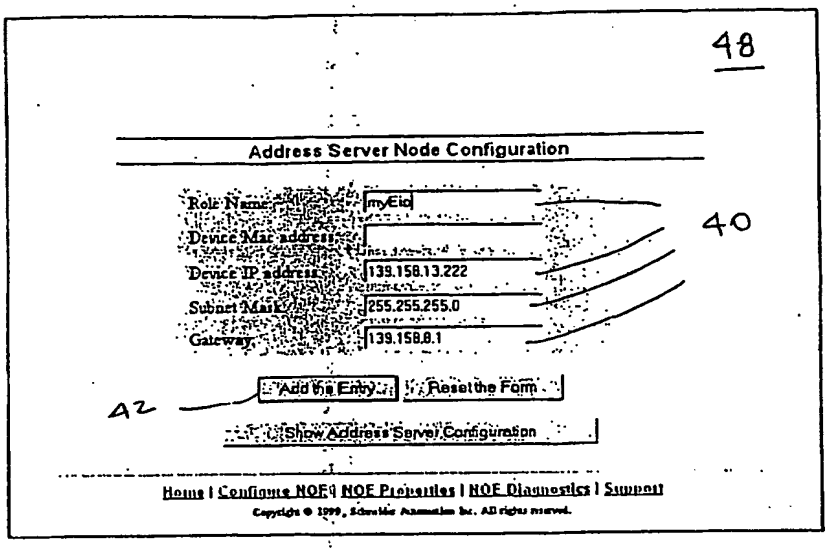
48
Address Server Node Configuration
Role Name:
myEio
Device Mac address:
Device IP address:
139.158.13.222
Subnet Mask:
255.255.255.0
Gateway:
139.158.8.1
40
42
Add the Entry
Reset the Form
Show Address Server Configuration
Home | Configure NOE | NOE Properties | NOE Diagnostics | Support
Copyright © 1999, Schneider Automation Inc. All rights reserved.

FIG. 15